Simplicity is Better: Revisiting Single Kernel PPI Extraction

Sung-Pil Choi
Information Technology Laboratory
Korea Institute of Science and Technology Information
spchoi@kisti.re.kr

Sung-Hyon Myaeng
Department of Computer Science
Korea Advanced Institute of Science and Technology
myaeng@kaist.ac.kr

Abstract

It has been known that a combination of multiple kernels and addition of various resources are the best options for improving effectiveness of kernel-based PPI extraction methods. These supplements, however, involve extensive kernel adaptation and feature selection processes, which attenuate the original benefits of the kernel methods. This paper shows that we are able to achieve the best performance among the state-of-the-art methods by using only a single kernel, convolution parse tree kernel. In-depth analyses of the kernel reveal that the keys to the improvement are the tree pruning method and consideration of tree kernel decay factors. It is noteworthy that we obtained the performance without having to use any additional features, kernels or corpora.

1 Introduction

Protein-Protein Interaction (PPI) Extraction refers to an automatic extraction of the interactions between multiple protein names from natural language sentences using linguistic features such as lexical clues and syntactic structures. A sentence may contain multiple protein names and relations, i.e., multiple PPIs. For example, the sentence in Fig.1 contains a total of six protein names of varying word lengths and three explicit interactions (relations). The interaction type between *phosphoprotein* and the acronym *P* in the parentheses is "*EQUAL*." A longer protein name *phosphoprotein of vesicular stomatitis virus* is related to *nucleocapsid protein* via "*INTERACT*" relation. Like the first PPI, *nucleocapsid protein* is equivalent to the abbreviated term *N*.

It is not straightforward to extract PPIs from a sentence or textual segment. There may be multiple protein names and their relationships, which are intertwined in a sentence. An interaction type may be expressed in a number of different ways.

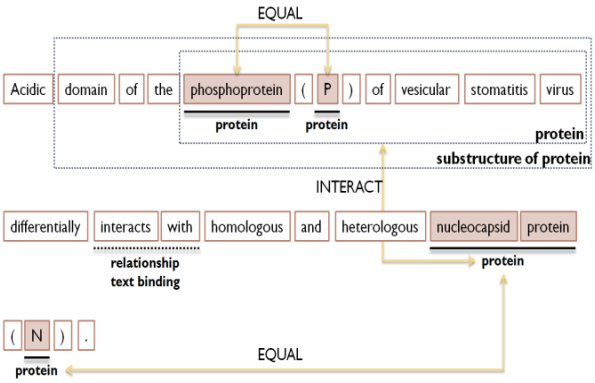

Figure 1. An example sentence containing multiple PPIs involving different names of varying scopes and relations[1]

A significant amount of efforts have been devoted to kernel-based approaches to PPI extractions (PPIE) as well as relation extractions[2] (Zhang et al., 2006; Pyysalo et al., 2008; GuoDong et al., 2007; Zhang et al., 2008; Airola et al., 2008; Miwa et al., 2009). They include word feature kernels, parse tree kernels, and graph kernels. One of the benefits of using a kernel method is that it can keep the original

[1] BioInfer, Sentence ID:BioInfer.d10.s0

[2] Relation extraction has been studied massively with the help of the ACE (www.nist.gov/tac) competition workshop and its corpora. The ACE corpora contain valuable information showing the traits of target entities (e.g., entity types, roles) for relation extraction in single sentences. Since all target entities are of the same type, protein name, in PPIE, however, we cannot use relational information that exists among entity types. This makes PPIE more challenging.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 206–214,
Beijing, August 2010

formation of target objects such as parse trees, not requiring extensive feature engineering for learning algorithms (Zelenko et al., 2003).

In an effort to improve the performance of PPIE, researchers have developed not only new kernels but also methods for combining them (GuoDong et al., 2007; Zhang et al., 2008; Airola et al., 2008; Miwa et al., 2009a; Miwa et al., 2009b). While the intricate ways of combing various kernels and using extra resources have played the role of establishing strong baseline performance for PPIE, however, they are viewed as another form of engineering efforts. After all, one of the reasons the kernel methods have become popular is to avoid such engineering efforts.

Instead, we focus on a state-of-the-art kernel and investigate how it can be best utilized for enhanced performance. We show that even with a single kernel, convolution parse tree kernel in this case, we can achieve superior performance in PPIE by devising an appropriate preprocessing and factor adjustment method. The keys to the improvement are tree pruning and consideration of a tree kernel decay factor, which are independent of the machine learning model used in this paper. The main contribution of our work is the extension and application of the particular convolution tree kernel method for PPIE, which gives a lesson that a deep analysis and a subsequent extension of a kernel for maximal performance can override the gains obtained from engineering additional features or combining other kernels.

The remaining part of the paper is organized as follows. In section 2, we survey the existing approaches. Section 3 introduces the parse tree kernel model and its algorithm. Section 4 explains the performance improving factors applied to the parse tree kernel. The architecture of our system is introduced in section 5. Section 6 shows the improvements in effectiveness in multiple PPI corpora and finally we conclude our work in section 7.

2 Related Work

In recent years, numerous studies have attempted to extract PPI automatically from text. Zhou and He (2008) classified various PPIE approaches into three categories: linguistic, rule-based and machine learning and statistical methods.

Linguistic approaches involve constructing special grammars capable of syntactically expressing the interactions in sentences and then applying them to the language analyzers such as part-of-speech taggers, chunkers and parsers to extract PPIs. Based on the level of linguistic analyses, we can divide the linguistic approaches into two categories: shallow parsing (Sekimizu et al., 1998; Gondy et al., 2003) and full parsing methods (Temkin & Gilder, 2003; Nikolai et al., 2004).

Rule-based approaches use manually defined sets of lexical patterns and find text segments that match the patterns. Blaschke et al. (1996) built a set of lexical rules based on clue words denoting interactions. Ono et al. (2001) defined a group of lexical and syntactic interaction patterns, embracing negative expressions, and applied them to extract PPIs from documents about "*Saccharomyces cerevisiae*" and "*Escherichia coli*". Recently, Fundel et al. (2007) proposed a PPI extraction model based on more systematic rules using a dependency parser.

Machine learning and statistical approaches have been around for a while but have recently become a dominant approach for PPI extraction. These methods involve building supervised or semi-supervised models based on training sets and various feature extraction methods (Andrade & Valencia, 1998; Marcotte et al., 2001; Craven & Kumlien, 1999). Among them, kernel-based methods have been studied extensively in recent years. Airola et al. (2008) attempted to extract PPIs using a graph kernel by converting dependency parse trees into the corresponding dependency graphs.

Miwa et al. (2009a) utilized multiple kernels such as word feature kernels, parse tree kernels, and even graph kernels in order to improve the performance of PPI extraction. Their experiments based on five PPI corpora, however, showed that combining multiple kernels gave only minor improvements compared to other methods. To further improve the performance of the multiple kernel system, the same group combined multiple corpora to exploit additional features for a modified SVM model (Miwa et al., 2009b). While they achieved the best performance in PPI extraction, it was possible only

with additional kernels and corpora from which additional features were extracted.

Unlike the aforementioned approaches trying to use all possible resources for performance enhancement, this paper aims at maximizing the performance of PPIE using only a single kernel without any additional resources. Without lowering the performance, we attempt to stick to the initial benefits of the kernel methods: *simplicity* and *modularity* (Shawe-Taylor & Cristianini, 2004).

3 Convolution Parse Tree Kernel Model for PPIE

The main idea of a convolution parse tree kernel is to sever a parse tree into its sub-trees and transfer it as a point in a vector space in which each axis denotes a particular sub-tree in the entire set of parse trees. If this set contains *M* unique sub-trees, the vector space becomes *M*-dimensional. The similarity between two parse trees can be obtained by computing the inner product of the two corresponding vectors, which is the output of the parse tree kernel.

There are two types of parse tree kernels of different forms of sub-trees: one is *SubTree Kernel* (*STK*) proposed by Vishwanathan and Smola (2003), and the other is *SubSet Tree Kernel* (*SSTK*) developed by Collins and Duffy (2001). In *STK*, each sub-tree should be a complete tree rooted by a specific node in the entire tree and ended with leaf nodes. All the sub-trees must obey the production rules of the syntactic grammar. Meanwhile, *SSTK* can have any forms of sub-trees in the entire parse tree given that they should obey the production rules. It was shown that *SSTK* is much superior to *STK* in many tasks (Moschitti, 2006). He also introduced a fast algorithm for computing a parse tree kernel and showed its beneficial effects on the semantic role labeling problem.

A parse tree kernel can be computed by the following equation:

$$K(T_1, T_2, \lambda, \sigma) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2, \lambda, \sigma) \quad (1)$$

where T_i is i^{th} parse tree and n_1 and n_2 are nodes in N_T, the set of the entire nodes of T. λ represents a tree kernel decay factor, which will be explained later, and σ decides the way the tree is severed. Finally $\Delta(n_1, n_2, \lambda, \sigma)$ counts the number of the common sub-trees of the two parse trees rooted by n_1 and n_2. Figure 2 shows the algorithm.

In this algorithm, the *get_children_number* function returns the number of the direct child nodes of the current node in a tree. The function named *get_node_value* gives the value of a node such as part-of-speeches, phrase tags and words. The *get_production_rule* function finds the grammatical rule of the current node and its children by inspecting their relationship.

```
1  FUNCTION delta(TreeNode n1, TreeNode n2, λ, σ)
2  n1 = one node of T1;  n2 = one node of T2;
3  λ = tree kernel decay factor;  σ = tree division me-
4  thod;
5  BEGIN
6     nc1 = get_children_number(n1);
7     nc2 = get_children_number(n2);
8     IF nc1 EQUAL 0 AND nc2 EQUAL 0 THEN
9        nv1 = get_node_value(n1);
10       nv2 = get_node_value(n2);
11       IF nv1 EQUAL nv2 THEN RETURN 1;
12    ENDIF
13    np1 = get_production_rule(n1);
14    np2 = get_production_rule(n2);
15    IF np1 NOT EQUAL np2 THEN RETURN 0;
16
17    IF np1 EQUAL np2 AND nc1 EQUAL 1
18          AND nc2 EQUAL 1 THEN
19       RETURN λ;
20    END IF
21
22    mult_delta = 1;
23    FOR I = 1 TO nc1
24       nch1 = I^th child of n1;   nch2 = I^th child of n2;
25       mult_delta = mult_delta ×
26                   (σ + delta(nch1, nch2, λ, σ));
27    END FOR
28    RETURN λ × mult_delta;
29 END
```

Figure 2. Δ (n_1, n_2, λ, σ) algorithm

4 Performance Improving Factors

4.1 Tree Pruning Methods

Tree pruning for relation extraction was firstly introduced by Zhang et al. (2006) and also referred to as "*tree shrinking task*" for removing less related contexts. They suggested five types of the pruning methods and later invented two more in Zhang et al. (2008). Among them, the path-enclosed tree (*PT*) method was shown to give the best result in the relation extraction task based on ACE corpus. We opted for this pruning method in our work.

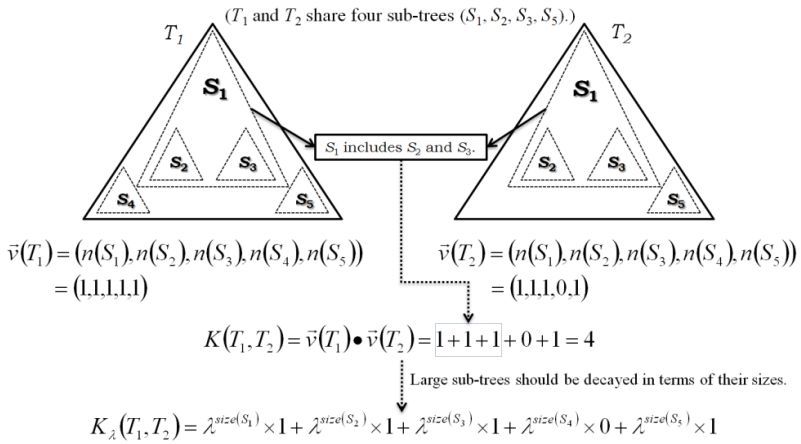

Figure 4. The effect of decaying in comparing two trees. $n(\cdot)$ denotes #unique subtrees in a tree.

Figure 3 shows how the *PT* method prunes a tree. To focus on the pivotal context, it preserves only the syntactic structure encompassing the two proteins at hand and the words in between them (the part enclosed by the dotted lines). Without pruning, all the words like *addition*, *increased* and *activity* would intricately participate in deciding the interaction type of this sentence.

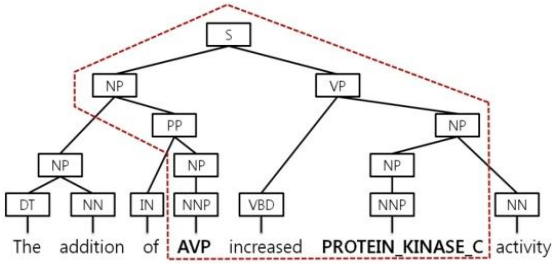

Figure 3. Path-enclosed Tree (PT) Method

Another important effect of the tree pruning is its ability to separate features when two or more interactions exist in a sentence. As in Figure 1, each interaction involves its unique context even though a sentence has multiple interactions. With tree pruning, it is likely to extract context-sensitive features by ignoring external features.

4.2 Tree Kernel Decay Factor

Collins and Duffy (2001) addressed two problems of the parse tree kernel. The first one is that its kernel value tends to be largely dominated by the size of two input trees. If they are large in size, it is highly probable for the kernel to accumulate a large number of overlapping counts in computing their similarity. Secondly, the kernel value of two identical parse trees can become overly large while the value of two different parse trees is much tiny in general. These two aspects can cause a trouble during a training phase because pairs of large parse trees that are similar to each other are disproportionately dominant. Consequently, the resulting models could act like nearest neighbor models (Collins and Duffy, 2001).

To alleviate the problems, Collins and Duffy (2001) introduced a scalability parameter called decay factor, $0 < \lambda \leq 1$ which scales the relative importance of tree fragments with their sizes as in line 33 of Fig. 2. Based on the algorithm, a decay factor decreases the degree of contribution of a large sub-tree exponentially in kernel computation. Figure 4 illustrates both the way a tree kernel is computed and the effect of a decay factor. In the figure, T_1 and T_2 share four common sub-trees (S_1, S_2, S_3, S_5). Let us assume that there are only two trees in a training set and only five unique sub-trees exist. Then each tree can be expressed by a vector whose elements are the number of particular sub-trees. Kernel value is obtained by computing the inner product of the two vectors. As shown in the figure, S_1 is a large sub-sub-trees, S_1, S_2 S_3, and S_4, two of which (S_2, and S_3) are duplicated in the inner product computation. It is highly probable for large sub-trees to contain many smaller sub-trees, which lead to an over-estimated similarity value between two parse trees. As mentioned above, therefore, it is necessary to rein those large sub-trees with respect to their sizes in computing kernel values by using decay factors. In this paper, we treat the decay factor as one of the important optimization parameters for a PPI extraction task.

5 Experimental Results

In order to show the superiority of the simple kernel based method using the two factors used in this paper, compared to the resent results for PPIE using additional resources, we ran a series of experiments using the same PPI corpora cited in the literature. In addition, we show that the method is robust especially for cross-corpus experiments where a classifier is trained and tested with entirely different corpora.

5.1 Evaluation Corpora

To evaluate our approach for PPIE, we used "*Five PPI Corpora*[3]" organized by Pyysalo et al. (2008). It contains five different PPI corpora: AImed, BioInfer, HPRD50, IEPA and LLL. They have been combined in a unified XML format and "*binarized*" in case of involving multiple interaction types.

	AIMed	BioInfer	HPRD50	IEPA	LLL
#Sentence	1,955	1,100	145	486	77
#Positive	1,000	2,534	163	335	164
#Negative	4,834	7,132	270	482	166

Table 1. Five PPI Corpora

Table 1 shows the size of each corpus in "*Five PPI Corpora*." As mentioned before, a sentence can have multiple interactions, which results in the gaps between the number of sentences and the sum of the number of instances. Negative instances have been automatically generated by enumerating sentences with multiple proteins but not having interactions between them (Pyysalo et al., 2008).

5.2 Evaluation Settings

In order to parse each sentence, we used Charniak Parser[4]. For kernel-based learning, we expanded the original *libsvm* 2.89[5] (Chang & Lin, 2001) so that it has two additional kernels including parse tree kernel and composite kernel[6] along with four built-in kernels[7]

Our experiment uses both macro-averaged and micro-averaged *F*-scores. Macro-averaging computes *F*-scores for all the classes individually and takes average of the scores. On the other hand, micro-averaging enumerates both positive results and negative results on the whole without considering the score of each class and computes total *F*-score.

In 10-fold cross validation, we apply the same split used in Airola et al., (2008), Miwa et al., (2009a) and Miwa et al., (2009b) for comparisons. Also, we empirically estimate the regularization parameters of SVM (*C*-values) by conducting 10-fold cross validation on each training data. We do not adjust the SVM thresholds to the optimal value as in Airola et al., (2008) and Miwa et al., (2009a).

5.3 PPI Extraction Performance

Table 2 shows the best scores of our system. The optimal decay factor varies with each corpus. In LLL, the optimal decay factor is 0.2[8] indicating that the shortage of data has forced our system to normalize parse trees more intensively with a strong decay factor in kernel computation in order to cover various syntactic structures.

	DF	*AC*	*ma*-P	*ma*-R	*ma*-F	$\sigma_{ma\text{-F}}$
A	0.6	83.6	72.8 (55.0)	62.1 (68.8)	**67.0** (60.8)	4.5 (6.6)
B	0.5	79.8	74.5 (65.7)	70.9 (71.1)	**72.6** (68.1)	2.7 (3.2)
H	0.7	74.5	75.3 (68.5)	71.0 (76.1)	**73.1** (70.9)	10.2 (10.3)
I	0.6	74.2	74.1 (67.5)	72.2 (78.6)	**73.1** (71.7)	6.0 (7.8)
L	0.2	82.2	83.2 (77.6)	81.2 (86.0)	**82.1** (80.1)	10.4 (14.1)

Table 2. The highest results of the proposed system w.r.t. decay factors. DF: Decay Factor, *AC*: accuracy, *ma*-F: macro-averaged F1, $\sigma_{ma\text{-F}}$: standard deviation of F-scores in CV. A:AIMed, B:BioInfer, H:HPRD50, I:IEPA, L:LLL. The numbers in parentheses refer to the scores of Miwa et al., (2009a).

Our system outperforms the previous results as in Table 2. Even using rich feature vectors including Bag-Of-Words and shortest path trees

[3] http://mars.cs.utu.fi/PPICorpora/eval-standard.html
[4] http://www.cs.brown.edu/people/ec/#software
[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[6] A kernel combining built-in kernels and parse tree kernel
[7] Linear, polynomial, radial basis function, sigmoid kernels

[8] It was determined by increasing it by 0.1 progressively through 10-fold cross validation.

generated from multiple corpora, Miwa et al., (2009b) reported 64.0% and 66.7% in AIMed and BioInfer, respectively. Our system, however, produced 67.0% in AIMed and 72.6% in BioInfer with a single parse tree kernel. We did not have to perform any intensive feature generation tasks using various linguistic analyzers and more importantly, did not use any additional corpora for training as done in Miwa et al., (2009b). While the performance differences are not very big, we argue that obtaining higher performance values is significant because the proposed system did not use any of the additional efforts and resources.

To investigate the effect of the scaling parameter of the parse tree kernel in PPI extraction, we measure how the performance changes as the decay factor varies (Figure 5). It is obvious that the decay factor influences the overall performance of PPI extraction. Especially, the *F*-scores of the small-scale corpora such as HPRD50 and LLL are influenced by the decay factor. The gaps between the best and worst scores in LLL and HPRD50 are 19.1% and 5.2%, respectively. The fluctuation in *F*-scores of the large-scale corpora (AIMed, BioInfer, IEPA) is not so extreme, which seems to stem from the abundance in syntactic and lexical forms that reduce the normalizing effect of the decay factor. The increase in the decay factor leads to the increase in the precision values of all the corpora except for LLL. The phenomenon is fairly plausible because the decreased normalization power causes the system to compute the tree similarities more intensively and therefore it classifies each instance in a strict and detailed manner. On the contrary, the recall values slightly decrease with respect to the decay factor, which indicates that the tree pruning (*PT*) has already conducted the normalization process to reduce the sparseness problem in each corpus.

Most importantly, along with tree pruning, decay factor could boost the performance of our system by controlling the rigidness of the parse tree kernel in PPI extraction.

Table 3 shows the results of the cross-corpus evaluation to measure the generalization power of our system as conducted in Airola et al., (2008) and Miwa et al., (2009a). Miwa et al., (2009b) executed a set of combinatorial experiments by mixing multiple corpora and presented their results. Therefore, it is not reasonable to compare our results with them due to the size discrepancy between training corpora. Nevertheless, we will compare our results with their approaches in later based on AIMed corpus.

As seen in Table 3, our system outperforms the existing approaches in almost all pairs of corpora. In particular, in the multiple corpora-based evaluations aimed at AIMed which has been frequently used as a standard set in PPI extraction, our approach shows prominent results compared with others. While other approaches showed the performance ranging from 33.3% to 60.8%, our approach achieved much higher scores between 55.9% and 67.0%. More specific observations are:

(1) Our PPIE method trained on any corpus except for IEPA outperforms the other approaches regardless of the test corpus only with a few exceptions with IEPA and LLL.

(2) Even when using LLL or HPRD50, two smallest corpora, as training sets, our system performs well with every other corpus for testing. It indicates that our approach is much less vulnerable to the sizes of training corpora than other methods.

(3) The degree of score fluctuation of our system across different testing corpora is much smaller than other regardless of the training data set. When trained on LLL, for example, the range for our system (55.9% ~ 82.1%) is smaller than the others (38.6% ~ 83.2% and 33.3% ~ 76.8%).

(4) The cross-corpus evaluation reveals that our method outperforms the others significantly. This is more visibly shown especially when the large-scale corpora (AIMed and BioInfer) are used.

(5) PPI extraction model trained on AIMed shows lower scores in IEPA and LLL as compared with other methods, which could trigger further investigation.

In order to convince ourselves further the superiority of the proposed method, we compare it with other previously reported approaches. Table 4 lists the macro-averaged precision, recall and *F*-scores of the nine approaches tested on AIMed. While the experimental settings are different as reported in the literature, they are quite close in terms of the numbers of positive and negative documents.

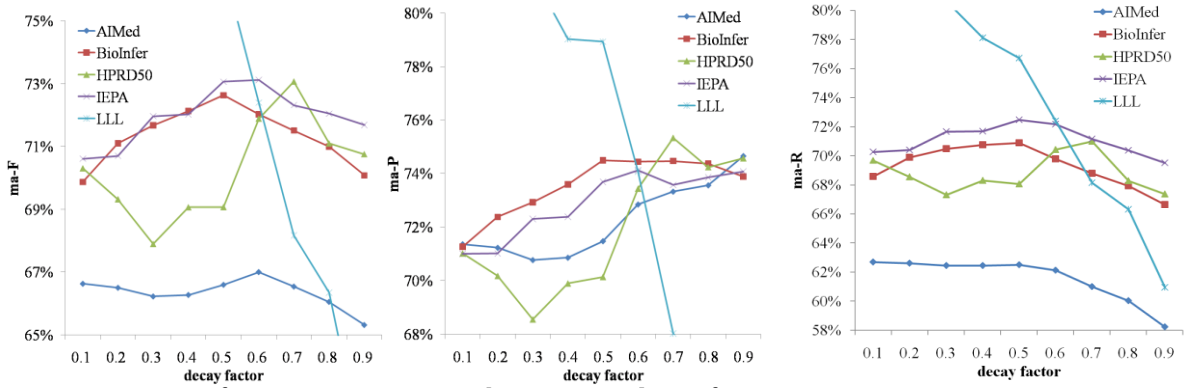

Figure 5. Performance variation with respect to decay factor in Five PPI Corpora. Macro-averaged F1 (left), Precision (middle), Recall (right) evaluated by 10-fold CV

Training corpora	Systems	*F*-Scores in the test corpora				
		AIMed	BioInfer	HPRD50	IEPA	LLL
AIMed	Our System	**67.0**	**64.2**	**72.9**	59.0	62.7
	(Miwa et al., 2009a)	60.8	53.1	68.3	**68.1**	73.5
	(Airola et al., 2008)	56.4	47.1	69.0	67.4	**74.5**
BioInfer	Our System	**65.2**	**72.6**	**71.9**	**72.9**	**78.4**
	(Miwa et al., 2009a)	49.6	68.1	68.3	71.4	76.9
	(Airola et al., 2008)	47.2	61.3	63.9	68.0	78.0
HPRD50	Our System	**63.1**	**65.5**	**73.1**	**69.3**	**73.7**
	(Miwa et al., 2009a)	43.9	48.6	70.9	67.8	72.2
	(Airola et al., 2008)	42.2	42.5	63.4	65.1	67.9
IEPA	Our System	**57.8**	**66.1**	66.3	73.1	78.4
	(Miwa et al., 2009a)	40.4	55.8	66.5	71.7	**83.2**
	(Airola et al., 2008)	39.1	51.7	**67.5**	**75.1**	77.6
LLL	Our System	**55.9**	**64.4**	**69.4**	**71.4**	82.1
	(Miwa et al., 2009a)	38.6	48.9	64.0	65.6	**83.2**
	(Airola et al., 2008)	33.3	42.5	59.8	64.9	76.8

Table 3. Macro-averaged *F*1 scores in cross-corpora evaluation. Rows and columns correspond to the training and test corpora, respectively. We parallel our results with other recently reported results. All the split methods in 10-fold CV are the same for fair comparisons.

As seen in the table, the proposed method is superior to all the others in *F*-scores. The improvement in precision (12.8%) is most significant, especially in comparison with the work of Miwa et al., (2009b), which used multiple corpora (AIMed + IEPA) for training and combined various kernels such as bag-of-words, parse trees and graphs. It is natural that the recall value is lower since a less number of patterns (features) must have been learned. What's important is that the proposed method has a higher or at least comparable overall performance without additional resources.

Our approach is significantly better than that of Airola et al., (2008), which employed two different forms of graph kernels to improve the initial model. Since they did not use multiple corpora for training, the comparison shows the direct benefit of using the extension of the kernel.

6 Conclusion and Future Works

To improve the performance of PPIE, recent research activities have had a tendency of increasing the complexity of the systems by combining various methods and resources. In this paper, however, we argue that by paying more

	POS	NEG	*ma*-P	*ma*-R	*m*a-F	σ_F
Our System	1,000	4,834	**72.8**	62.1	**67.0**	**4.5**
(Miwa et al., 2009b)	1,000	4,834	60.0	**71.9**	65.2	
(Miwa et al., 2009a)	1,000	4,834	58.7	66.1	61.9	7.4
(Miwa et al., 2008)	1,005	4,643	60.4	69.3	61.5	
(Miyao et al., 2008)	1,059	4,589	54.9	65.5	59.5	
(Giuliano et al., 2006)	-	-	60.9	57.2	59.0	
(Airola et al., 2008)	1,000	4,834	52.9	61.8	56.4	5.0
(Sætre et al., 2007)	1,068	4,563	64.3	44.1	52.0	
(Erkan et al., 2007)	951	4,020	59.6	60.7	60.0	
(Bunescu & Mooney, 2005)	-	-	65.0	46.4	54.2	

Table 4. Comparative results in AIMed. The number of positive instances (POS) and negative instances (NEG) and macro-averaged precision (*ma*-P), recall (*ma*-R) and *F*1-score (*ma*-F) are shown.

attention to a single model and adjusting parameters more carefully, we can obtain at least comparable performance if not better.

This paper indicates that a well-tuned parse tree kernel based on decay factor can achieve the superior performance in PPIE when it is preprocessed by the path-enclosed tree pruning method. It was shown in a series of experiments that our system produced the best scores in single corpus evaluation as well as cross-corpora validation in comparison with other state-of-the-art methods. Contribution points of this paper are as follows:

(1) We have shown that the benefits of using additional resources including richer features can be obtained by tuning a single tree kernel method with tree pruning and decaying factors.

(2) We have newly found that the decay factor influences precision enhancement of PPIE and hence its overall performance as well.

(3) We have also revealed that the parse tree kernel method equipped with decay factors shows superior generalization power even with small corpora while presenting significant performance increase on cross-corpora experiments.

As a future study, we leave experiments with training the classifier with multiple corpora and deeper analysis of what aspects of the corpora gave different magnitudes of the improvements.

Acknowledgment

We want to thank the anonymous reviewers for their valuable comments. This work has been supported in part by KRCF Grant, the Korean government.

Reference

Airola, A., Pyysalo, S., Bjorne, J., Pahikkala, T., Ginter, F. & Salakoski, T. (2008). All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. BMC Bioinformatics, 9(S2), doi:10.1186/1471-2105-9-S11-S2.

Andrade, M. A. & Valencia, A. (1998). Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. Bioinformatics, 14(7), 600-607.

Blaschke, C., Andrade, M., Ouzounis, C. & Valencia, A. (1999). Automatic extraction of biological information from scientific text: protein-protein interactions. Proc. Int. Conf. Intell. Syst. Mol. Biol., (pp. 60-67).

Bunescu, R., Ge, R., Kate, R., Marcotte, E., Mooney, R., Ramani, A. & Wong, Y. (2005). Comparative Experiments on Learning Information Extractors for Proteins and their Interactions. Artif. Intell. Med., Summarization and Information Extraction from Medical Documents, 33, 139-155

Collins, M. & Duffy, N. (2001). Convolution Kernels for Natural Language. NIPS-2001, (pp. 625-632).

Craven, M. & Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. Proceedings of the 7th International conference on intelligent systems for molecular biology, (pp.77-86), Heidelberg, Germany.

Ding, J., Berleant, D., Nettleton, D. & Wurtele, E. (2002). Mining MEDLINE: abstracts, sentences, or phrases?. Proceedings of PSB'02, (pp. 326-337)

Erkan, G., Ozgur, A., & Radev, D. R., (2007). Semi-supervised classification for extracting protein in-

teraction sentences using dependency parsing. In EMNLP 2007.

Fundel, K., Küffner, R. & Zimmer, R. (2007). RelEx – Relation extraction using dependency parse trees. Bioinformatics, 23, 365-371.

Giuliano, C., Lavelli, A., Romano, L., (2006). Exploiting Shallow Linguistic Information for Relation Extraction From Biomedical Literature. Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics.

Gondy, L., Hsinchun C. & Martinez Jesse D. (2003). A shallow parser based on closed-class words to capture relations in biomedical text. J. Biomed. Informatics. 36(3), 145-158.

GuoDong, Z., Min, Z., Dong, H. J. & QiaoMing, Z. (2007). Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, (pp. 728–736)

Marcotte, E. M., Xenarios, I. & Eisenberg D. (2001). Mining literature for protein-protein interactions. Bioinformatics, 17(4), 359-363.

Miwa, M., Sætre, R., Miyao, Y. & Tsujii J. (2009a). Protein-protein interaction extraction by leveraging multiple kernels and parsers. International Journal of Medical Informatics, 78(12), e39-e46.

Miwa, M., Sætre, R., Miyao, Y. & Tsujii J. (2009b). A Rich Feature Vector for Protein-Protein Interaction Extraction from Multiple Corpora. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, (pp. 121-130)

Miwa, M., Sætre, R., Miyao, Y., Ohta, T., & Tsujii, J. (2008). Combining multiple layers of syntactic information for protein-protein interaction extraction. In Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008), (pp. 101–108)

Miyao, Y., Sætre, R., Sagae, K., Matsuzaki, T., & Tsujii, J. (2008). Task-oriented evaluation of syntactic parsers and their representations. Proceedings of the 45th Meeting of the Association for Computational Linguistics (ACL'08:HLT).

Moschitti, A. (2006). Making tree kernels practical for natural language learning. Proceedings of EACL'06, Trento, Italy.

Nikolai, D., Anton, Y., Sergei, E., Svetalana, N., Alexander, N. & llya, M. (2004). Extracting human protein interactions from MEDLINE using a full-sentence parser. Bioinformatics, 20(5), 604-611.

Ono, T., Hishigaki, H., Tanigam, A. & Takagi, T. (2001). Automated extraction of information on protein-protein interactions from the biological literature. Bioinformatics, 17(2), 155-161.

Pyysalo, S., Airola, A., Heimonen, J., Björne, J., Ginter, F. & Salakoski, T. (2008). Comparative analysis of five protein-protein interaction corpora. BMC Bioinformatics, 9(S6), doi:10.1186/1471-2105-9-S3-S6.

Sætre, R., Sagae, K., & Tsujii, J. (2007). Syntactic features for protein-protein interaction extraction. In LBM 2007 short papers.

Sekimizu, T., Park H. S. & Tsujii J. (1998). Identifying the interaction between genes and gene products based on frequently seen verbs in MEDLINE abstracts. Workshop on genome informatics, vol. 9, (pp. 62-71).

Shawe-Taylor, J., Cristianini, N., (2004). Kernel Methods for Pattern Analysis, Cambridge University Press.

Temkin, J. M. & Gilder, M. R. (2003). Extraction of protein interaction information from unstructured text using a context-free grammar. Bioinformatics, 19(16), 2046-2053.

Vishwanathan, S. V. N., Smola, A. J. (2003). Fast Kernels for String and Tree Matching. Advances in Neural Information Processing Systems, 15, 569-576, MIT Press.

Zhang, M., GuoDong, Z. & Aiti, A. (2008). Exploring syntactic structured features over parse trees for relation extraction using kernel methods. Information Processing and Management, 44, 687-701

Zhang, M., Zhang, J., Su, J. & Zhou, G. (2006). A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, (pp.825-832).

Zhou, D. & He, Y. (2008). Extracting interactions between proteins from the literature. Journal of Biomedical Informatics, 41, 393-407.